DBSJ Japanese Journal
Vol. 15-J, Article No. 9, March 2017

リンケージポリシ執行を保証するレコードリンケージ

Record Linkage with Record-wise Linkage Policy

海寶 貴人♡ 陸 文杰◇ 天笠 俊之♠
佐久間 淳♣

Takahito KAIHO　Wenjie LU
Toshiyuki AMAGASA　Jun SAKUMA

個人情報を個別に集積した複数のデータベースに対して同一人物のレコードを連結するレコードリンケージは，新たな知見の獲得に貢献するが，個人の望まない，あるいは意図しない連結はプライバシ侵害を招く．本研究では，個人が主体的に任意の **2** データベース間のレコードリンケージの可否についてプライバシポリシを定め，個人が望まないレコードリンケージは実行できないような準同型暗号に基づく暗号プロトコルを提案する．論文では，N データベース中の任意の **2** データベース間の連携の拒否を指定するホワイトリスト方式と，同様に任意の **2** データベース間の連携を拒否するブラックリスト方式の二通りのプライバシポリシを定式化し，そのようなポリシに従うレコードリンケージの制御を実現する暗号プロトコルの正しさとセキュリティを示す．また実験により，データベース数 **50** 個のレコードリンケージについては，各データ登録者に対してホワイトリスト方式では **1** 秒未満，ブラックリスト方式では約 **23** 秒でプロトコルを実施できることを示した．

Record linkage is the calculation that associate records among multiple databases. This method contributes on analyzing big data and learning hiding knowledge. Meanwhile, record linkage may violate the privacy of user who send his data to the database by relating to record. In this paper, we define linkage policy which decides whether executes joining linkage between arbitrary two different databases in multiple databases and present private preserving record linkage protocol by enforcing linkage policy using homomorphic encryption. We represent the linkage policy as an undirected graph and formalize security model. We show security and correctness of our protocol following such model and policy. We achieve the security among stakeholder by using Paillier cryptosystem. Also, we attempt experiment of an execution time of our protocol. We show that our protocol can execute in at most 23 seconds on 50 databases regardless of the number of users.

♡ 筑波大学システム情報工学研究科博士前期課程
takahito@mdl.cs.tsukuba.ac.jp
◇ 筑波大学システム情報工学研究科博士後期課程
riku@mdl.cs.tsukuba.ac.jp
♠ 正会員　筑波大学システム情報系
amagasa@cs.tsukuba.ac.jp
♣ 筑波大学システム情報系
jun@cs.tsukuba.ac.jp

1. はじめに

インターネットの技術発達により，サイトに個人情報を登録することでサービスを受けられるシステムが増加している．例として，サイトに登録された住所へユーザが注文した商品を配達する電子商取引のサービスが挙げられる．サイト内の個人情報は，データの集積を行うデータベースにレコード形式で保管されている．個人情報を保管した複数のデータベースについて，同一のユーザに関するレコードを連結し共通点や関連性などを分析することによって新たなサービスの創出や新たな知識の発見につながることが期待されている．例として，以下のようなサービスが挙げられる．

1. インターネットの利用時，検索サイトで検索したワードにしたがって，電子通販サイトや各種サイト内の広告バナーに，検索ワードの関連商品を表示する．
2. 保険会社が，自社の利用者のデータと病院が管理するデータを組み合わせることで，利用者に最適なプランを紹介する．

1 の例はすでに実現されている例であり，2 はこれから予想されるサービスの例である．

複数のデータベースに対するデータの分析を行うときに有用な技術の一つとしてレコードリンケージが挙げられる．レコードリンケージとは，異なるデータベースから共通の要素を持つデータレコードを発見し，そのレコードの組を結合する技術である．

より多くのデータベースの共通の要素を連結するレコードリンケージを実施することで，データに対してさらに詳しい分析を行える．日本に限っていえば，2015 年に成立した改正個人情報保護法では，匿名加工情報の同意なき移転，流通を認めている．しかし，無制限にレコードリンケージを実施した場合にデータ登録者のプライバシが侵害されるリスクがある．具体的には，データ登録者が特定の事業者に対してのみ連結を許していた情報が異なる事業者の保持する情報に意図せず連結される可能性がある．(2) のサービスが実現した場合を例に考える．保険会社と病院のデータをリンケージした後，最寄りの病院の推薦を目的としてデータの分析者が病院と SNS サイトのデータベースをリンケージした場合， SNS サイト事業者が別途保険会社からデータ提供をうけたときに，病院のデータを手掛かりにして保険会社と SNS サイトのデータをリンケージできる．このときデータ登録者は，自身の疾患，保険料に関する情報など，保険会社と病院にのみ公開したい情報が SNS サイトのデータと結合することでに SNS 事業者に知られる恐れがある．

近年，データ登録者のプライバシを考慮したレコードリンケージについての研究が行われているが，いずれの手法においても，d 個のデータベースのリンケージによって得られる d^2 個のリレーションの管理や，データ登録者の意思に対する配慮がなされていない．本研究では，これらを考慮したレコードリンケージをコントロールするためのプロトコルを提案する．

1.1 関連研究：プライバシ保護レコードリンケージ

レコードリンケージとは，複数のデータベースに対して同一の人物についてのデータを結合する技術であり，複数のデータベースに対してデータマイニングや統計値の分析などを行うときに有用である．現在，わが国において官公庁や企業間で統一されている識別子は少ないが，マイナンバー法の施行によりレコードリンケージを用いた情報分析への期待が高まっている．

データ登録者のプライバシを保護しながらレコードリンケージを行うプライバシ保護レコードリンケージについて紹介する．プライバシ保護レコードリンケージでは，識別子やデータそのものの暗号化，匿名化を行うことでユーザのデータを公開せずに安全なレコードリンケージを達成することを目的とする．具体的には，偽のデータを含めたデータレコードの暗号化を行う手法 [1]，フォ

利用者 ID	メールアドレス	購入金額	住所
001	one@ex.com	¥5,900	東京都○○区……
002	two@ex.ac.jp	¥8.500	茨城県□□市……
003	three@ex.co.jp	¥14,200	千葉県××市……
004	four@ex.com	¥7,300	埼玉県△△市……

表 1: 例 : 電子商取引サイトの Database が保管するデータの一部

口座 ID	電話番号	預金額	住所
001	080-www-wwww	¥332,000	東京都○○区……
002	090-xxx-xxxx	¥169,000	茨城県□□市……
003	090-yyy-yyyy	¥562,000	千葉県××市……
004	080-zzz-zzzz	¥234,000	埼玉県△△市……

表 2: 例 : 銀行の Database が保管するデータの一部

ネティックコードを用いてデータの変換を行う手法 [2]，Bloom Filter を用いて暗号化した文字列が一致するか否かを確かめる手法 [3]，データをブロックごとに分け各ブロックの差分プライバシを考慮する手法 [4] などが挙げられる．論文 [5] では，元データの改変，データに付与されるラベルの暗号化，データの曖昧化を行うことでデータ登録者の情報を匿名化し，データ登録者のプライバシを保護する手法が現在までの研究として示されている．

1.2 貢献

本研究が既存の手法と異なる点は大きく二つある．一つ目は，データの分析結果の相違である．データの匿名化を行う手法では，データの本来のデータとは異なるデータを用いてリンケージを行うため正確なデータ分析を実施することが困難である．本研究では，暗号上の加減算が可能な準同型暗号を用いてデータを加工せずにリンケージを行うため，正しい分析結果を得ることができる．二つ目は，システム管理者に対してポリシの執行を強制できることである．既存の研究ではデータ登録者のデータに対する処理のみを行っており，データ登録者の意思を考慮していない．本研究では，ユーザのポリシを暗号化することで，システム管理者に悪意がある場合でも安全にレコードリンケージを行うことができる．

本研究では，データ登録者のプライバシに配慮しながら正確な情報を用いたレコードリンケージを実現するため，データ登録者がデータ分析者に対してプライバシポリシを執行，すなわち連結の可否を任意の二つのデータベースに対して指定できるプロトコルを提案する．ポリシを保護しながらデータ結合を行うことを可能とするため，準同型暗号の一種である Paillier 暗号を導入した．また，提案したプロトコルの実行時間を検証し，本手法が現実的な時間で実施可能であることを示す．

2. 問題設定

2.1 ステークホルダ

本研究のデータの操作に関連するステークホルダを以下に示す．

- Database … Data Contributor からデータを受け取り, 管理を行う. Database は複数存在しており, Database 一つ一つに保持者が存在すると仮定する. Database の保持者の例として, 銀行, 商取引サイト, 病院, IC カードの管理会社などが挙げられる. これらのサイトには, 表 1, 表 2 のような形式で個人ごとにデータが保管されている.
- Data Contributor (以下, DC) … Database にレコード形式でデータを登録する. DC は複数存在している. 各 DC には一意な識別子が割り当てられており, すべての Database にデータを識別子とともに登録していると仮定する. 1 レコードは 1 人の個人情報を表す.
  表 1, 表 2 を結合することで得られる結果の一部を表 3 に示す. 以上の表から各個人の電子商取引サイトの購入総額と銀行の預金額が明らかになる. このとき, 預金額と購入金額総計を同時に公開することで, DC は商取引サイトに銀行の預金額を把握される. 商取引サイトに銀行のデータを閲覧されたくない DC は, 結合を拒否する意向を示すことで自身のデータを結合結果から取り除くことができる. 具体的には, DC は任意の二つの Database からなる各組に対して, 自分自身のデータレコードの結合を許可するか拒否するかを決定できる. 結合を「許可」するか「拒否」するかを示す意向を本研究ではポリシとよぶ.
- Analyst … 任意の二つのデータベースや属性を指定し, レコードリンケージを実施する結合クエリを発行し, 結合表を得て統計的な知識の獲得や分析を行う. Analyst は Database である場合も, 第三者である場合も想定する. Analyst は, Linkage Manager に対してクエリを送信し, Database 内のデータの結合表を Linkage Manager から受け取る. 例として, 電子商取引サイトでの購入総額と銀行の預金額に相関性があるかどうかを分析したい Analyst は, 表 3 を得るために Linkage Manager へ該当する二つのデータベースの購入金額と預金額についてリンケージ依頼をクエリとして送信する.
- Linkage Manager (以下, LM) … Analyst からのリクエストに応じ, 指定された Database 間に存在するデータの結合が可能かどうかを各 DC ごとに調べ, 結合表を返す.

ID	購入金額（商取引）	預金額（銀行）
001	¥5,900	¥332,000
002	¥8,500	¥169,000
003	¥14,200	¥562,000
004	¥7,300	¥234,000

表 3: 例：表結合の結果の一部

i	$\in \mathcal{I}$	各 DC の識別子
c	$\in \mathbb{N}$	DC の総数
j	$\in \mathcal{J}$	各 Database の識別子
d	$\in \mathbb{N}$	Database の総数
$l_{st}(i)$	$\in \{0,1\}$	DC i がもつ Database(s,t) のポリシ
x_{ij}	$\in \mathcal{X}$	DC i が Database j に登録するデータ
q		Analyst が LM へ送信するクエリ

表 4: ノーテーション

システムの全体像, 動作の様子を 図 1 に, ノーテーションを表 4 に記す.

2.2 ポリシの表現方法について

各 DC が所有するポリシを表現するために, 頂点はシステム内の Database に, 辺は Database 間のポリシに対応した無向グラフを用いる. 本研究では, この無向グラフをポリシグラフとよぶ. c 人の DC が d 個の Database それぞれにデータを登録しているとき, ポリシグラフは DC 一人につき一つ生成され, システム全体に c 個のポリシグラフが存在している. Database を区別するために, 各 Database には番号 $j \in \mathcal{J}$ が割り振られている. ポリシグラフの表現方法として, 以下の 2 種類を挙げる.

2.2.1 ホワイトリスト方式

ポリシグラフの辺が連携許可を表す場合, この方式をホワイトリスト方式とよぶ. ホワイトリスト方式のポリシグラフの頂点間に辺が存在しない場合, 対応する Database 間の連結は拒否されているものとみなす. ホワイトリストのポリシグラフを表現する例として, 図 2 を挙げる. 頂点の番号は各 Database に割り振られている番号を表す. 図 2 の例では, Database 1（電子商取引サイトの Database） と 2（コンビニエンスストアの Database）, および Database 3（銀行の Database） と 5（電子マネーカードの Database） の連携のみが許可されており, 他の任意の二つの Database 間は連携拒否である状態を表す.

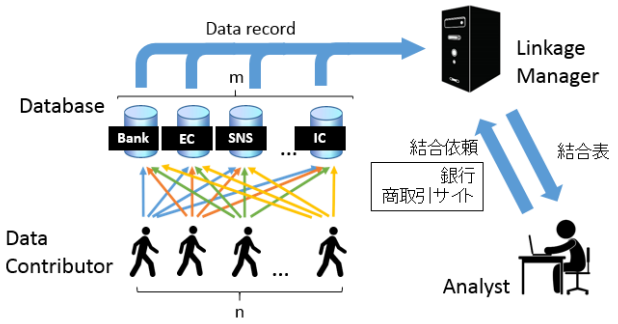

図 1: システムの全体像

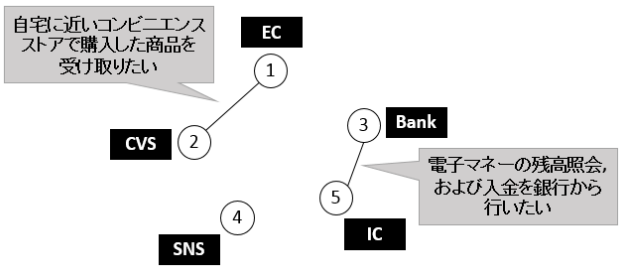

図 2: ホワイトリストの例：ホワイトリストの辺を実線で示す．

2.2.2 ブラックリスト方式

ポリシグラフの辺が連携拒否を表す場合，この方式をブラックリスト方式とよぶ (図 3)．ブラックリスト方式のポリシグラフの頂点間に辺が存在しない場合，対応する Database 間に関するポリシは，これまでの辺の連結によって決定される．図 4 は連結の履歴によって現在許可できる連結が異なる例を示している．図上段のポリシグラフは Database 1 と 3 の連結を拒否するブラックリストである．Analyst が 1 回目のクエリで Database 1 と 2 の連結を行う場合をケース (1,2)，Database 2 と 3 の連結を行う場合をケース (2,3) とし，以上の二つのケースについて考える．図下段の二つのポリシグラフは，ケースごとのデータ連結後のグラフである．ケース (1,2) 以降のクエリにおいて，Database 2 と 3 の連結を行うと，(1,2) の連結を経由して，ブラックリストである Database 1 と 3 が連結されるため，Database 2 と 3 の連結は拒否されるべきである．同様に，ケース (2,3) においても，これ以降のクエリで Database 1 と 2 の連結は拒否されるべきである．具体例を図 5 に示す．

各 DC が個別に Database 間の連結をコントロールする状況において，特定の Database 間の結合を許可するホワイトリスト方式よりも，結合を拒否するブラックリスト方式がより現実的である．一方，ブラックリスト方式の連結許可は過去の連結の履歴によって変化するため，ホワイトリスト方式と比較してポリシが不安定である．

2.3 セキュリティモデル

ステークホルダ間において閲覧可能なデータ，閲覧不可能なデータを明確にするため，当事者とデータの関係を明記したセキュリティモデルを表 5 に示す．

データ x_{ij} の閲覧が可能なステークホルダは，データの保持者である i-th DC および DC のデータを管理する j-th Database である．LM はデータレコードの結合表のみを返すため，DC が保持するデータを閲覧する必要がない．Analyst は結合表に応じてデータ閲覧の可否が異なり，クエリの対象となる二つの Database 間のポリシが許可ならばリンケージの結果であるデータを閲覧できる．

ポリシグラフ G_i の閲覧が可能なステークホルダは，ポリシグラフの保持者である i-th DC のみである．DC 以外のステークホルダがポリシグラフを閲覧できる場合，悪意のあるステークホルダが DC に無断でポリシを変更することで，拒否のポリシが指定されている二つの Database に対してもレコードが結合できる可能性がある．

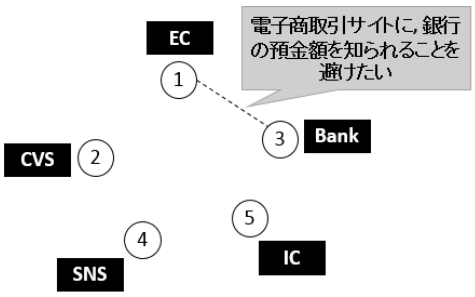

図 3: ブラックリストの例：ブラックリストの辺を破線で示す．各 Database は honest である．

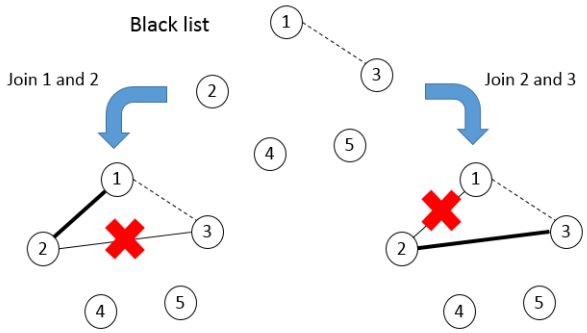

図 4: 連結履歴によってポリシが相違する例：ブラックリストの辺を破線で，クエリ履歴を太線で示す．

クエリ履歴 Q_i の閲覧が可能なステークホルダは，クエリの対象となるデータを持つ i-th DC，およびクエリの連結を行う LM である．ブラックリスト方式において，LM は各クエリに対応する二つの Database の連結が許可されるか否かを逐一判断するため，クエリ履歴を閲覧する必要がある．Analyst は結合表に応じてクエリ履歴閲覧の可否が異なり，クエリの対象となる二つの Database 間のポリシが許可ならばその結果としてクエリ履歴を閲覧できる．

2.4 問題定義

グラフの表現方式別に問題定義を示す．

2.4.1 ホワイトリスト方式

Analyst は LM へ二つの Database の番号 s, t を送る．LM は，DC が保持するホワイトリストをもとに，$l_{st}(i) = 1$ である DC のレコードを連結した表を Analyst へ送る．これら一連の処理をセキュリティモデルに違反せずに実施する．

2.4.2 ブラックリスト方式

Analyst は LM へ二つの Database の番号 s, t を送る．LM は，DC が保持するブラックリスト，および Analyst がそれまでに実施したクエリの履歴をもとに，Database s, t の連結を行ってもブラックリストが連結されない DC のレコードを連結した表を Analyst へ送る．これら一連の処理をセキュリティモデルに違反せずに実施する．

3. 要素技術

本章では，セキュリティモデルに基づいたプロトコルを実現するための要素技術を導入する．

3.1 reachability matrix

ブラックリスト方式では，クエリごとにポリシが許可であるか否かをクエリ履歴を用いて調べる必要がある．この処理を行うために

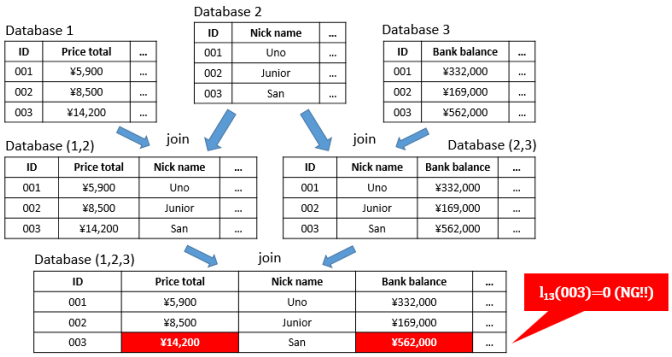

図 5: Database(1,2) と Database(2,3) の連携から Database(1,3) が連携される例

	DC	Database	LM	Analyst
x_{ij}	○	○	×	△
G_i	○	×	×	×
Q_i	○	×	○	△

表 5: セキュリティモデル : クエリ履歴は Analyst がクエリを行った後に連携が許可された Database の組が辺に対応する無向グラフを表す. 「○」は, 自分自身がデータの保持者であるとき閲覧可能であることをあらわす. 「△」は, 対象の DC のポリシが許可である場合のみ閲覧可能であることをあらわす. 「×」は閲覧不可能であることをあらわす.

ポリシグラフの各頂点の連結を reachability matrix で表現する. グラフに対する reachability matrix を生成する際には隣接行列を用いる. 隣接行列はサイズが $|V|\times|V|$ である行列である. 頂点 s, 頂点 t をつなぐ辺が存在する場合には, 隣接行列の (s,t) 要素は 1, そうでなければ 0 をとる. 隣接行列の定義から, 重みなしのグラフの隣接行列 $\mathbf{A}$ のべき乗 $\mathbf{A}^n$ の (i,j) 成分は, 頂点 i から頂点 j への長さ n のパスが存在するか否かをあらわす. 頂点数 $|V|$ のグラフにおける reachability matrix $\mathbf{R}$ は以下の式であらわせる [7].

$$\mathbf{R} = \mathbf{A} + \mathbf{A}^2 + \cdots + \mathbf{A}^{|V|} \tag{1}$$

$\mathbf{R}$ の (i,j) 成分が 1 以上であれば, i から j へのパスが存在するといえる.

3.2 Paillier 暗号

Paillier 暗号 [8] は加法準同型性をみたす公開鍵暗号方式である. 加法準同型暗号とは, 暗号化された整数どうしの加算を平文同様に行うことが可能である暗号である. 乱数 $r\in\mathbb{Z}_n$ と公開鍵 pk を用いて平文 $m\in\mathbb{Z}_n$ を暗号化する処理を $c=\mathsf{Enc}(m;r)$ と記述する. また, 秘密鍵 sk を用いて暗号文 c を復号する処理を $m=\mathsf{Dec}(c)$ と記述する. Paillier 暗号は以下の性質をもつ.

$$\mathsf{Enc}(m_1;r_1)\mathsf{Enc}(m_2;r_2) = \mathsf{Enc}(m_1+m_2;r_1+r_2) \tag{2}$$

$$\mathsf{Enc}(m_1;r_1)^{m_2} = \mathsf{Enc}(m_1m_2;r_1m_2) \tag{3}$$

以降, 暗号化, および復号の記述を簡略化するため, それぞれ $\mathsf{Enc}(\cdot)$, $\mathsf{Dec}(\cdot)$ と記述する. また, 準同型暗号上の加算を「$\oplus$」, 暗号文と平文との乗算を「$\otimes$」と表記する.

4. 提案方式

本節では, ポリシグラフの表現別に定式化したアルゴリズムを示す.

4.1 ホワイトリスト方式

ホワイトリスト方式のポリシグラフを W_i とすると, 対応するプロトコルは以下の手順によって実現される.

- i-th DC は以下の操作を行う.
  1. データの連携を許可したい Database の組をすべて選択し, W_i を構成する. 構成する W_i 内の部分集合はすべてクリークとなるよう加工する.
  2. W_i の 0, 1 を反転した W_i' を生成する.
  3. W_i' から, 隣接行列 $\mathbf{A}_i^{W'}$ を構成する.
  4. 各 Database に対し, データレコード $(i, x_{ij}, (\mathbf{A}_i^{W'}))$ を登録する.
- データレコードの登録後, Analyst が次の操作を任意の回数行う.
  1. LM に対して, $q=(s,t)$ を問合わせる.
  2. クエリの結果として $\mathfrak{Q}_i = \mathsf{Enc}(x_{is}+a_i)|\mathsf{Enc}(x_{it}+a_i)$ を得る. ここで $x|y$ は, データ x, y の結合をあらわす. また, a_i は条件に応じて次の値をとる. r は $\mathbb{Z}_n$ 上の乱数を示す.

$$a_i = \begin{cases} 0 & (s,t)\in W_i \\ r & otherwise \end{cases}$$

ホワイトリストの部分集合がクリークでないとき, 不自然なポリシが発生する. 例として, Database(1,2), および (2,3) のポリシが連結許可であるホワイトリストを考える. このとき, Database(1,3) のポリシは禁止と設定されているが, Database(1,2), および (2,3) のリンケージを行ったときに Database (1,3) のデータは連結できるため, ホワイトリストの観点から Database (1,3) のポリシが禁止であることは不自然である. したがって, ホワイトリスト方式のグラフはクリークとなるよう加工する必要がある.

以降, ホワイト方式のプロトコルを実現するアルゴリズム Linkage_White および, Analyst がシステムに送信したクエリを入力として各 DC の結合表を出力するアルゴリズム Query_White について述べる.

4.1.1 Linkage_White

Algorithm 1 Linkage_White

Input of DC: W_i

Output of Analyst: $x_{is}|x_{it}$ $(\forall i\ l_{st}(i)=1)$

1: Analyst は, 公開鍵 pk, 秘密鍵 sk を生成する.
2: i-th DC は, W_i の隣接行列の 0 と 1 を反転させたグラフ W_i' を生成する.
3: i-th DC は以下の処理を行う.
4: **for** $j=1$ **to** d **do**
5: $\quad$ i-th DC は, pk を用いて $\mathsf{Enc}(\mathbf{A}_i^{W'})$, および $\mathsf{Enc}(x_{ij})$ を j-th Database へ登録する.
6: **end for**
7: **SendQuery**: Analyst は, $\{\mathfrak{Q}_1,\cdots,\mathfrak{Q}_c\}\leftarrow$ Query_White(s,t,pk) を得る.
8: i-th DC のポリシが許可ならば, Analyst は sk を用いて $x_{is}|x_{it}=\mathsf{Dec}(\mathfrak{Q}_i)$ を得る.
9: 新たな分析を望む Analyst は **SendQuery** 以降の処理をふたたび実行する.

アルゴリズム Linkage_White を **Algorithm 1** に記す. LM が行う演算はすべて暗号上で行われるため, DC のポリシを閲覧できない. また, Analyst は, LM から最終的な計算結果を得るのみであるため, 結合表以外の情報を閲覧できない.

4.1.2 Query_White

Algorithm 2 Query_White

Input: $s \in \mathbb{N}$, $t \in \mathbb{N}$, pk

Output: $\{\mathfrak{Q}_1, \cdots, \mathfrak{Q}_c\}$

1: Database s および t は, $\mathsf{Enc}(\mathbf{A}^{W'})$, および $\mathsf{Enc}(x_{1s}), \cdots, \mathsf{Enc}(x_{cs})$, $\mathsf{Enc}(x_{1t}), \cdots, \mathsf{Enc}(x_{ct})$ を LM へわたす.
2: LM は, 乱数 r を生成する.
3: **for** $i = 1$ **to** c **do**
4: $\quad \mathsf{Enc}(a_i) \leftarrow r \otimes \mathsf{Enc}((\mathbf{A}_i^{W'})_{st})$
5: $\quad \mathfrak{Q}_i \leftarrow \mathsf{Enc}(x_{is} + a_i) | \mathsf{Enc}(x_{it} + a_i)$
6: **end for**
7: LM は, $\{\mathfrak{Q}_1, \cdots, \mathfrak{Q}_c\}$ を Analyst へわたす.

アルゴリズム Query_White を **Algorithm 2** に記す. Query_White では, Analyst が Database の識別子をあらわす整数の組 (s,t) を LM へ入力する. LM は, 与えられた s, t, および W' の隣接行列に基づいて $\mathsf{Enc}(a_i)$ を計算し, 表結合 $\mathsf{Enc}(x_{is} + a_i) | \mathsf{Enc}(x_{it} + a_i)$ を生成する. Paillier 暗号は加法について準同型性をもつため, LM は x_i, および a_i に関する情報を得ずに $\mathsf{Enc}(x_i + a_i) = \mathsf{Enc}(x_i) \oplus \mathsf{Enc}(a_i)$ を計算できる. i-th DC のポリシが許可であるとき, $a_i = 0$ であるため $x_{is} | x_{it} = \mathsf{Dec}(\mathfrak{Q}_i)$ が成立する. 一方, i-th DC のポリシが拒否であるとき, $\mathbb{Z}_n$ 上の乱数を r_1, r_2 として, 表結合の結果は $r_1 | r_2 = \mathsf{Dec}(\mathfrak{Q}_i)$ とあらわせるため, Analyst は正しい結合結果を得られない.

4.2 ブラックリスト方式

ブラックリスト方式のポリシグラフを B_i とすると, 対応するプロトコルは以下の手順によって実現される.

- i-th DC は以下の操作を行う.
  1. B_i から隣接行列 $\mathbf{A}_i^B$ を構成する.
  2. 各 Database に対し, データレコード $(i, x_{ij}, (\mathbf{A}_i^B))$ を登録する.
- データレコードの登録後, 隣接行列が零行列であるグラフ Q_i を生成し, Analyst が次の操作を任意の回数行う.
  1. LM に対して, $q = (s,t)$ を問合わせる.
  2. クエリの結果として, $\mathfrak{Q}_i = \mathsf{Enc}(x_{is} + a_i) | \mathsf{Enc}(x_{it} + a_i)$ を得る. ここで $x|y$ は, データ x, y の結合をあらわす. また, a_i は条件に応じて次の値をとる. r は $\mathbb{Z}_n$ 上の乱数を示す.

$$a_i = \begin{cases} 0 & \mathbf{R}_{st} = 0 \quad {}^\forall(s,t) \in B_i \\ r & otherwise \end{cases}$$

  3. クエリの結果を得た後, 結合結果が正しければ Q_i を更新する.

LM は reachability matrix を用いてクエリのたびにポリシの連結が許可されているかどうか確認し, ポリシが侵害される場合にはデータの連結を防ぐ必要がある. そこで, ブラックリスト方式では, ホワイトリスト方式で用いたアルゴリズムに reachability matrix を計算する処理を加えてアルゴリズムの設計を行う. 以降, ブラック方式のプロトコルを実現するアルゴリズム Linkage_Black, Query_Black について述べる.

Algorithm 3 Linkage_Black

Input of DC: B_i

Output of Analyst: $x_{is} | x_{it}$ $({}^\forall i\ l_{st}(i) \neq 0)$

1: Analyst は, 公開鍵 pk, 秘密鍵 sk を生成する.
2: i-th DC は以下の処理を行う.
3: **for** $j = 1$ **to** d **do**
4: $\quad$ i-th DC は, pk を用いて $\mathsf{Enc}(\mathbf{A}_i^B)$, および $\mathsf{Enc}(x_{ij})$ を j-th Database へ登録する.
5: **end for**
6: LM は零行列 $Q_1, \cdots, Q_c$ を生成する.
7: **SendQuery**: Anlyst は, $\{\mathfrak{Q}_1, \cdots, \mathfrak{Q}_c\} \leftarrow$ Query_Black(s, t, pk) を得る.
8: i-th DC のポリシが許可ならば, Analyst は sk を用いて $x_{is} | x_{it} = \mathsf{Dec}(\mathfrak{Q}_i)$ を得る.
9: 正しい結果が得られたならば, Analyst は Database(s,t) の連携を許可であるとみなし, $(Q_i)_{st} \leftarrow 1$, $(Q_i)_{ts} \leftarrow 1$ を実施する.
10: 新たな分析を望む Analyst は **SendQuery** 以降の処理をふたたび実行する.

4.2.1 Linkage_Black

Linkage_Black のアルゴリズムを **Algorithm 3** に記す. 行列 Q_i は各 DC に対するクエリの履歴をあらわす. Query を実施するごとに, 各 DC が連携許可か否かを確認し, 連携が許可されているならば 9 行目でクエリ履歴の更新を行う.

4.2.2 Query_Black

Query_Black のアルゴリズムを **Algorithm 4** に記す. ここで, $\mathbf{0}$ は零行列, I は単位行列である. 3 行目から 6 行目の処理では, クエリ履歴をあらわす隣接行列 Q の reachability matrix R をもとめる. 頂点 s, t を連結する際にブラックリストが連結するか否かを確認するために 11 行目の処理を行う.

ここで, reachability matrix の生成を行う第 3 章の式 (1) は次のかたちに書き直すことができる.

$$\mathbf{R}_k = \mathbf{A} \times (I + \mathbf{R}_{k-1}) \quad (4)$$
$$\mathbf{R}_0 = \mathbf{0} \quad (5)$$

$|V|$ 個の点からなる隣接行列 $\mathbf{A}$ であるグラフの reachability matrix は $\mathbf{R}_{|V|}$ に相当する. 式 (4), (5) を用いることで, reachability matrix 内の加乗算の演算回数を $O(|V|^2)$ 回から $O(|V|)$ 回へ削減できる.

4.3 計算量, 通信ラウンドの比較

ホワイトリスト方式とブラックリスト方式の演算回数を比較する. ホワイトリスト, ブラックリストの両方式において暗号文と平文の乗算回数は $O(1)$, 準同型暗号上の加算回数は最悪で $O(d^2)$ 回である. また, ブラックリスト方式の reachability matrix を生成する処理に用いる平文の加乗算の回数は, 行列積をもとめる処理から $O(cd^3)$ 回である.

以上のことから, Database の数, および DC の数が増えるごとに, ホワイトリスト方式による処理時間とブラックリスト方式による処理時間の差が大きくなることが考えられる.

Algorithm 4 Query_Black

Input: $s \in \mathbb{N}$, $t \in \mathbb{N}$, pk, $\mathbf{Q}_i$,

Output: $\{\mathfrak{Q}_1, \cdots, \mathfrak{Q}_c\}$

1: Database s および t は，$\mathsf{Enc}(\mathbf{A}^B)$，および $\mathsf{Enc}(x_{1s}), \cdots, \mathsf{Enc}(x_{cs})$, $\mathsf{Enc}(x_{1t}), \cdots, \mathsf{Enc}(x_{ct})$ を LM へわたす．
2: LM は次の処理を行う．$\mathbf{R}' \leftarrow \mathbf{0}$
3: **for** $i = 1$ **to** c **do**
4: **for** $j = 1$ **to** d **do**
5: $\mathbf{R}_i \leftarrow \mathbf{Q}_i \cdot (I + \mathbf{R}'_i)$
6: $\mathbf{R}'_i \leftarrow \mathbf{R}_i$
7: **end for**
8: LM は，点集合 $S_i := \{p | (R_i)_{sp} > 0\}$ を生成する．
9: LM は，点集合 $T_i := \{q | (R_i)_{qt} > 0\}$ を生成する．
10: LM は，乱数 r を生成する．
11: $\mathsf{Enc}(a_i) \leftarrow r \otimes \bigoplus_{s' \in S_i, t' \in T_i} \mathsf{Enc}((\mathbf{A}_i^B)_{s't'})$
12: $\mathfrak{Q}_i \leftarrow \mathsf{Enc}(x_{is} + a_i) | \mathsf{Enc}(x_{it} + a_i)$
13: **end for**
14: LM は，$\{\mathfrak{Q}_1, \cdots, \mathfrak{Q}_c\}$ を Analyst へわたす．

5. 実験

提案する各プロトコルについて，時間の計測を行う．実験に用いた計算機の CPU は intel Core i7-2600 3.5GHz，メモリ量は 8GB である．プログラムは，ローカル環境で C++を用いて実装した．

5.1 実験設定

DC 一人のポリシグラフに対する，プロトコル内のアルゴリズム Query_White，Query_Black にかかる時間，およびグラフの隣接行列を暗号化する処理にかかる時間を計測する．Linkage_White および Linkage_Black では，DC がグラフの隣接行列を暗号化する処理が実行時間の大半を占めている．ブラックリスト方式については，Query_Black において reachability matrix を生成する処理にかかる時間を加えて計測する．d を 5 から 50 まで 5 刻みで 10 回異なるポリシグラフに対して同一のクエリを送信するプロトコルを実施し，該当するプロトコルにかかる時間の平均を計測した．各 LinkageProtocol において Query を呼び出す回数は 1 回である．また，Paillier 暗号の鍵長は 1024bit とした．

5.2 結果と考察

$c = 1$ のときの Query_White，および Query_Black の実行時間の計測結果を図 6 に示す．Query_White における計算回数が d に依存しない一方，Query_Black では，サイズ $d \times d$ の reachability matrix を生成するため，d が増加するごとに指数関数的に計算時間が増加する．c 人の DC についてレコードリンケージを行う場合，Query_White，Query_Black の所要時間は，図 6 に示した各プロトコルの所要時間と DC の人数 c の積であらわせる．

LinkageProtocol_White，LinkageProtocol_Black の両プロトコルにおいて，サイズ $d \times d$ の行列を暗号化する処理がもとめられる．サイズ $d \times d$ の行列の暗号化に要する時間を図 7 に示す．$d = 5$ の際は 1 sec 未満で，$d = 50$ の際は約 23 sec で行列の暗号化を行える．図 7 より，行列の暗号化の処理時間は d に応じて指数関数的に増大することがわかる．行列の暗号化の処理はプロトコルの最初に実行するため，システム内の Database の個数が増加しない限り一回のみで済む．また DC 一人一人が行列の暗号化を行うため，プロトコル全体で行列の暗号化にかかる処理は図 7 に示す時間で完了すると考えてよい．

6. おわりに

本論文では，データ登録者が任意のデータベース間のプライバシポリシを保持することで，データ登録者が望まないレコードリンケージが実行できないような準同型暗号に基づいた暗号プロトコルを，グラフで表現したポリシの方式に応じて二通り提案した．一つ目の手法は連結許可のポリシを定めるホワイトリスト方式であり，隣接行列のポリシのマスクを行うことでポリシの改ざんを防ぐことを実現した．二つ目の手法は連結拒否のポリシを定めるブラックリスト方式であり，グラフの隣接行列を用いた reachability test を実施することでポリシの秘匿を実現した．プライバシに対するデータ登録者の需要からブラックリスト方式がより実用的である．一方，ホワイトリスト方式はブラックリスト方式に比べ計算量が小さいため高速なレコードリンケージを行える．また，実行時間の観点から二手法のプロトコルを提案した．また，クエリ単位の実行時間についても現実的に実行可能であることを示した．

本研究の主な発展的課題を二点述べる．一つ目はホワイトリストとブラックリストの混合方式の実現である．ホワイトリスト方式では分析者が柔軟な分析をできない一方，ブラックリスト方式では LM の計算が複雑であり，データ登録者が意識してポリシを指定できない場合がある．そこで混合方式では，データ登録者にとって確保したいデータは保持しつつ，Analyst にとって柔軟な分析が実現することを目指す．二つ目は，ポリシに属性を付与する方式である．データ登録者，また分析者の属性ごとにポリシを指定することで，データ分析者がその用途に応じて適切なレコードリンケージを行えることを目指す．この方式の実現のために，属性ベース暗号を用いる予定である．属性ベース暗号とは，複数の公開鍵，秘密鍵を設け，各鍵が一対一で対応している公開鍵暗号方式である．例としては，医療調査者や予備校の関係者など，分析が制限されている sensitive なデータに対して分析を行えるよ

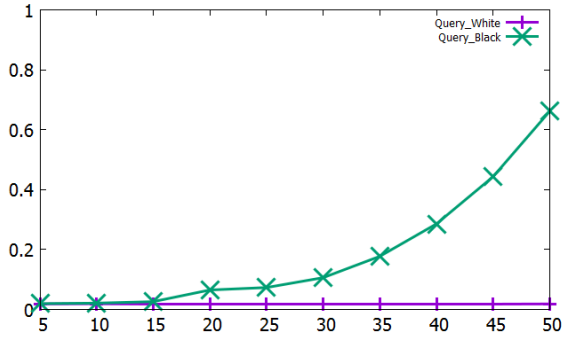

図 6: Query_White，Query_Black の実行時間 [sec]

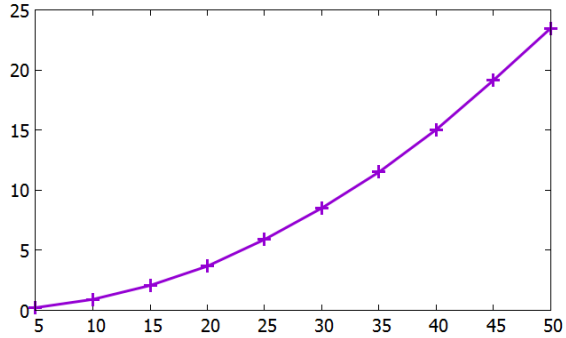

図 7: サイズ $d \times d$ の行列の暗号化に要する時間 [sec]

うな者のみが，適切なデータベースに対してアクセスができるようなことを可能にする．

[謝辞]

本研究は，JST CREST「ビッグデータ統合利活用のための次世代基盤技術の創出・体系化」領域におけるプロジェクト，科学研究費 24680015，および文部科学省／理化学研究所「実社会ビッグデータ利活用のためのデータ統合・解析技術の研究開発」の助成を受けました．

[文献]

[1] Alexandros Karakasidis and Vassilios S. Verykios, "Secure blocking+ secure matching= secure record linkage", Journal of Computing Science and Engineering, pp. 223–235, 2011.

[2] Alexandros Karakasidis and Vassilios S. Verykios, "Privacy preserving record linkage using phonetic codes", In: Informatics, 2009. BCI'09. Fourth Balkan Conference in IEEE, pp. 101–106, 2009.

[3] Rainer Schnell, Tobias Bachteler, et al, "Privacy-preserving record linkage using Bloom filters", BMC medical informatics and decision making, 2009.

[4] Mehmet Kuzu, et al, "Efficient privacy-aware record integration", In: Proceedings of the 16th International Conference on Extending Database Technology. pp. 167–178. ACM, 2013.

[5] Hye-Cheng Kum and Ashok Krishnamurthy, et al, "Privacy preserving interactive record linkage (PPIRL)", Journal of American Medical Informatics Association, 2013.

[6] Nigel P Smart and Frederik Vercauteren, "Fully homomorphic SIMD operations." Designs, codes and cryptography, 71(1) pp.57–81, 2014.

[7] Jean-Paul Tremblay and G. A. Cheston, "Data Strucutres oftware Development in an Object-Oriented Environment. Java Edition", Prentice Hall, 2003.

[8] Pascal Paillier, "Public-key cryptosystems based on composite degree residuosity classes", In Proceedings of the 17th international conference on Theory and application of cryptographic techniques, EUROCRYPT'99, pp. 223–238, Berlin, Heidelberg, Springer-Verlag, 1999.

海寶 貴人 Takahito KAIHO
筑波大学システム情報工学研究科博士前期課程在学．2015 年 3 月中央大学理工学部情報工学科卒業．

陸 文杰 Wenjie LU
筑波大学システム情報工学研究科博士後期課程在学．2016 年 3 月筑波大学システム情報工学研究科博士前期課程修了．

天笠 俊之 Toshiyuki AMAGASA
筑波大学計算科学研究センター／システム情報系准教授．1999 年群馬大学大学院工学研究科修了．博士（工学）．奈良先端科学技術大学院大学情報科学研究科助手，筑波大学大学院システム情報工学研究科講師を経て，現在に至る．データベース，データマイニング等の研究に従事．情報処理学会，日本データベース学会，ACM 各会員．電子情報通信学会，IEEE 各シニア会員．

佐久間 淳 Jun SAKUMA
筑波大学システム情報系教授．2003 年 3 月東京工業大学大学院総合理工学研博士後期課程修了．博士 (工学)．同年 4 月日本アイ・ビー・エム株式会社入社，東京基礎研究所に配属．2004 年 7 月，東京工業大学総合理工学研究科助手，2007 年 4 月同助教，2009 年 4 月，筑波大学大学院システム情報工学研究科准教授，2016 年 4 月同教授，現在に至る．機械学習と知識発見，セキュリティとプライバシの研究に従事．